RAIRO Operations Research
RAIRO Oper. Res. **36** (2002) 21-36
DOI: 10.1051/ro:2002003

ON THE HARDNESS OF APPROXIMATING THE UET-UCT SCHEDULING PROBLEM WITH HIERARCHICAL COMMUNICATIONS

Evripidis Bampis[1], R. Giroudeau[1] and J.-C. König[2]

Communicated by Philippe Chrétienne

Abstract. We consider the unit execution time unit communication time (UET-UCT) scheduling model with hierarchical communications [1], and we study the impact of the hierarchical communications hypothesis on the hardness of approximation. We prove that there is no polynomial time approximation algorithm with performance guarantee smaller than 5/4 (unless $\mathcal{P} = \mathcal{NP}$). This result is an extension of the result of Hoogeveen *et al.* [6] who proved that there is no polynomial time ρ-approximation algorithm with $\rho < 7/6$ for the classical UET-UCT scheduling problem with homogeneous communication delays and an unrestricted number of identical machines.

Keywords: Scheduling, hierarchical communications, non-approximability.

Mathematics Subject Classification. 90B35.

1. Introduction

We consider an extension of the classical scheduling problem with communication delays [2] which takes into account hierarchical communications [1] (this

Received September, 1999.

[1] Laboratoire de Méthodes Informatiques (LaMI), Université d'Évry-Val-d'Essonne, UMR 8042 du CNRS, 523 place des Terrasses, Immeuble ÉVRY-II, 91000 Évry, France; e-mail: bampis@lami.univ-evry.fr, giroudeau@lami.univ-evry.fr

[2] LIRMM, Université de Montpellier II, UMR 5506 du CNRS, 161 rue Ada, 34392 Montpellier Cedex 5, France; e-mail: konig@lirmm.fr

© EDP Sciences 2002

extension is motivated by the advance of parallel architectures comprising several identical multiprocessors). We are given m multiprocessors machines (or clusters) that are used to process n precedence constrained tasks. Each machine (cluster) comprises several identical parallel processors. A couple (c_{ij}, ϵ_{ij}) of communication delays is associated to each arc (i, j) between two tasks in the precedence graph. In what follows, c_{ij} (resp. ϵ_{ij}) is called intercluster (resp. interprocessor) communication, and we consider that $c_{ij} \geq \epsilon_{ij}$. If tasks i and j are executed on different machines, then j must be processed at least c_{ij} time units after the completion of i. Similarly, if i and j are executed on the same machine but on different processors then the processing of j can only start ϵ_{ij} units of time after the completion of i. However, if i and j are executed on the same processor then j can start immediately after the end of i. The communication overhead (intercluster or interprocessor delay) does not interfere with the availability of the processors and all processors may execute other tasks. Our goal is to find a feasible schedule of the tasks minimizing the *makespan*, *i.e.* the time at which the last task of the precedence graph finishes its execution.

Notice that the hierarchical model that we consider here is a generalization of the classical scheduling model with communication delays [2, 3]. Consider for instance that for every arc (i, j) of the precedence graph we have $c_{ij} = \epsilon_{ij}$. In that case the hierarchical model is exactly the classical scheduling communication delays model. It is then clear that every negative ($\mathcal{NP}$-hardness, or non-approximability) result known for the scheduling problem with communication delays is also valid for the more general hierarchical model. Hoogeveen *et al.* [6] proved that, unless $\mathcal{P} = \mathcal{NP}$, the well known scheduling problem with communication delays $\bar{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$ does not possess a polynomial time approximation algorithm with ratio less than 7/6. In our context, this problem can be viewed as follows: there is an unrestricted number of monoprocessor machines, all tasks have unit execution times and every intercluster communication costs one unit of time, *i.e.* $c_{ij} = 1$. Given that there is only one processor per machine we have $\epsilon_{ij} = 0$.

In this paper we answer to the following natural question: *What is the impact of the hierarchical communication hypothesis on the non-approximability of the related scheduling problem?* In order to answer to this question, we consider here the simplest extension of the problem studied by Hoogeveen *et al.* ($\bar{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max}$) in which every machine comprises two identical processors and $\epsilon_{ij} = 0$. We denote this extension as $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\max}$. In [1], it has been proved that this problem is polynomial when task duplication is allowed. Here, we prove that, unless $\mathcal{P} = \mathcal{NP}$, there is no ρ-approximation algorithm with $\rho < \frac{5}{4}$ for the hierarchical problem. The proof is based on a reduction from a special case of SAT.

In Section 2, we give the definition of the considered variant of SAT and we prove that it is an $\mathcal{NP}$-complete problem. In Section 3, we present the non-approximability result which is based on the well known Impossibility Theorem [4]. Finally in Section 4, we propose a polynomial time algorithm for the problem of

deciding whether a precedence graph can be executed within 3 units of time in the hierarchical communication model.

2. Preliminaries

In order to prove that $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{\max} = 4$ is an $\mathcal{NP}$-complete problem, we use a reduction from a variant of the well known SAT problem [4]. We will call this variant the *One-in-*$(2,3)SAT(2,\bar{1})$ problem that we will denote as Π_1. Π_1 is a restricted variant of SAT with clauses of length two and three, where each variable x occurs three times: two of these occurrences are unnegated (in the form of literal x) and one negated (in the form of literal $\bar{x}$). One of the unnegated occurrences is in a clause of length three and the other in a clause of length two. The literal $\bar{x}$ is necessarily in a clause of length two different from the clause of size two in which the literal x occurs. Thus the clause $(x \vee \bar{x})$ cannot exist in any instance of Π_1. Moreover, for every two literals $\bar{x}$ and y, occurring in a same clause of length two, the corresponding literals x and y must occur in two different clauses of length three. We are interested in the existence or not of a truth assignment in which every clause has *exactly* one true literal.

Formally, *One-in-*$(2,3)SAT(2,\bar{1})$ can be stated as follows:

Instance of problem Π_1:

- Let $\mathcal{V} = \{x_1, \ldots, x_n\}$ be a set of variables and $\bar{\mathcal{V}} = \{\bar{x}_1, \ldots, \bar{x}_n\}$ the set of negated variables with $x_i \in \mathcal{V}$.
- Let $\mathcal{C} = \{C_1, \ldots, C_j, C_{j+1}, \ldots, C_q\}$ be a set of clauses where $\forall i, 1 \leq i \leq j, C_i \in (\mathcal{V} \times \bar{\mathcal{V}})$ and $\forall i, (j+1) \leq i \leq q$ (with $q = \frac{4n}{3}$), $C_i \in (\mathcal{V})^3$, and such that every variable $x_i \in \mathcal{V}$ occurs two times unnegated and one time negated: $\forall x_i \in \mathcal{V}, \exists i_1, i_2, i_3$ such that

$$\forall x_i \in \mathcal{V}, \begin{cases} occur(x_i; C_{i_1}) = 1 \text{ and } occur(x_i; C_{i_2}) = 1, & 1 \leq i_1 \leq j,\ j+1 \leq i_2 \leq q \\ occur(\bar{x}_i; C_{i_3}) = 1 & 1 \leq i_3 \leq j,\ i_3 \neq i_1 \end{cases}$$

  where $occur(x_i; C_{i_k})$ is a function that gives the number of times where variable x_i occurs in the clause C_{i_k}.

  In addition, if ($x_i \in C_k$ and $\bar{x}_{i'} \in C_k, 1 \leq k \leq j$) then ($x_i \in C_l$) and ($x_{i'} \in C_r$), for some $l \neq r$ and $(j+1) \leq l, r \leq q$.

Question. Is there a truth assignment for $I : \mathcal{V} \to \{0,1\}$ such that every clause in $\mathcal{C}$ has exactly a true literal?

In order to illustrate Π_1, we consider the following example:

Example 2.1. The following logic formula is a valid instance of Π_1: $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge (\bar{x}_0 \vee x_3) \wedge (\bar{x}_3 \vee x_0) \wedge (\bar{x}_4 \vee x_2) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_5 \vee x_1) \wedge (\bar{x}_2 \vee x_5)$.

The answer to Π_1 is *yes*. It is sufficient to choose $x_0 = 1$, $x_3 = 1$ and $x_i = 0$ for $i = \{1, 2, 4, 5\}$. This gives a truth assignment satisfying the formula, and there is exactly one true literal in every clause.

Theorem 2.2. Π_1 *is an* $\mathcal{NP}$*-complete problem.*

In order to prove the $\mathcal{NP}$-completeness of Π_1, we use a reduction from a restricted version of a well-known $\mathcal{NP}$-complete problem, the *Monotone-one-in-three*-3SAT problem [4], that we call *Monotone-one-in-three*-3$SAT^*(\geq 2)$ and we denote by Π_2. First, we prove that Π_2 is an $\mathcal{NP}$-complete problem.

Let us, first recall the definition of *Monotone-one-in-three*-3SAT problem.

Instance of problem *Monotone-one-in-three*-3SAT:

- Let $\mathcal{V} = \{x_1, \ldots, x_n\}$ be a set of n variables.
- Let $\mathcal{C} = \{C_1, \ldots, C_m\}$ be a collection of clauses over $\mathcal{V}$ such that every clause has size three and contains only unnegated variables.

Question. Is there a truth assignment for $\mathcal{V}$ such that every clause in $\mathcal{C}$ has exactly one true literal?

Π_2 is defined in the same way as *Monotone-one-in-three*-3SAT, except that in Π_2 every variable occurs at least twice and there are no two occurrences of a same variable in the same clause. The problem Π_2 is an $\mathcal{NP}$-complete problem: since *Monotone-one-in-three*-3SAT is $\mathcal{NP}$-complete, without loss of generality we can suppose that each variable x_i occurs at least twice otherwise it is sufficient to add a copy of the clause in which it belongs. In addition, a literal x_i cannot occur twice in the same clause since then we can easily reduce the size of the instance: a variable x_i should have the value $false$ and the third variable in the clause the value $true$.

Theorem 2.3. Π_1 *is an* $\mathcal{NP}$*-complete problem.*

Proof. It is easy to see that $\Pi_1 \in \mathcal{NP}$.

Our proof is based on a reduction from Π_2.

Given any instance π^* of the problem Π_2, we construct an instance π of Π_1 in the following way:

- If a variable x_i occurs $k_i \geq 2$ times in π^*, then we rename the j^{th} occurrence ($1 \leq j \leq k_i$) of x_i by introducing a new variable $x_{i_{(j-1)}}$. Let $\mathcal{V}'$ be the set of new variables obtained in this way. In every clause of π^*, we rename the occurring variables in a greedy manner and we complete the corresponding instance π by adding the following clauses of length two: $(x_{i_{(j-1)}} \vee \bar{x}_{i_{(j \bmod k_i)}}), \forall i, \forall j, 1 \leq j \leq k_i$. Let $\mathcal{C}'$ be the set of the obtained clauses.

  It is now easy to verify that every instance π of Π_1 obtained by the above construction respects the following two properties:

Property 1. Every variable of $\mathcal{V}'$ occurs three times in π. More precisely, every variable occurs:

- two times unnegated, and more precisely one time in a clause of length three and one time in a clause of length two;
- one time negated in a clause of length two different from the clause in which its unnegated occurrence appears.

Property 2. The variables of $\mathcal{V}'$ occurring in a same clause of length two are such that their unnegated occurrences belong to disjoint clauses of length three.

- Suppose that there is a truth assignment $I : \mathcal{V} \to \{0,1\}$, such that each clause in $\mathcal{C}$ has exactly one true literal. In the following, we will prove that there is a truth assignment $I' : \mathcal{V}' \to \{0,1\}$ such that each clause in $\mathcal{C}'$ has also exactly one true literal.
  If we take $I'(x_{i_{(j-1)}}) = I(x_i), \forall i, j, 1 \leq j \leq k_i$, we can see first that all clauses of length three become true and each of them has exactly one true literal, since all clauses of length three in $\mathcal{C}$ respect this property.
  In addition, it is clear that every clause of length two is satisfied and only one literal in each of them is true.
  Consequently, if π^* is satisfied then π is also satisfied.
- Conversely, assume that there is a truth assignment $I' : \mathcal{V}' \to \{0,1\}$ such that each clause of $\mathcal{C}'$ has exactly one true literal. In the following, we will prove that there is a truth assignment $I : \mathcal{V} \to \{0,1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal. Because of the form of the clauses of length two $(x_{i_{(j-1)}} \vee \bar{x}_{i_{(j \bmod k_i)}}), \forall i, \forall j, 1 \leq j \leq k_i$ and given that I' is such that there is exactly one true literal in every clause, we can conclude that all the variables x_{i_k}, for every fixed i and any k, have the same assignment in I'. In order to find a truth assignment for $\mathcal{C}$, it is sufficient to put $I(x_i) = I'(x_{i_0})$ for every i. Clearly, this assignment respects the desired property of the uniqueness of a true literal per clause.
  Consequently, if π is satisfied then π^* is also satisfied.

The above transformation can be computed in polynomial time and so Π_1 is $\mathcal{NP}$-complete. □

Remark 2.4. By a careful reading of the previous reduction one can see that an instance in which there is a variable x for which the clause $(x \vee \bar{x})$ appears in $\mathcal{C}$ is not a valid instance of Π_1. This remark is essential for the proof of Theorem 3.1.

3. The non-approximability result

Theorem 3.1. *The problem of deciding whether an instance of* $\bar{P}(P2)|$ prec; $(c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{\max}$ *has a schedule of length at most four is* $\mathcal{NP}$*-complete.*

Proof. It is easy to see that $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{\max} = 4 \in \mathcal{NP}$. The rest of the proof is based on a reduction from Π_1.

Let n be the number of variables in the logic formula. The clauses are labeled from 1 to q.

Given an instance π^* of Π_1, we construct an instance π of the problem $\bar{P}(P2)|$ prec; $(c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{\max} = 4$, in the following way:

- For each variable $x \in \mathcal{V}$, we introduce five variable-tasks $\hat{x}, x, x', \bar{x}$ and $\bar{x}'$. The precedence constraints between these tasks are the following:
  $\hat{x} \to x, \hat{x} \to x', \hat{x} \to \bar{x}, \hat{x} \to \bar{x}'$.

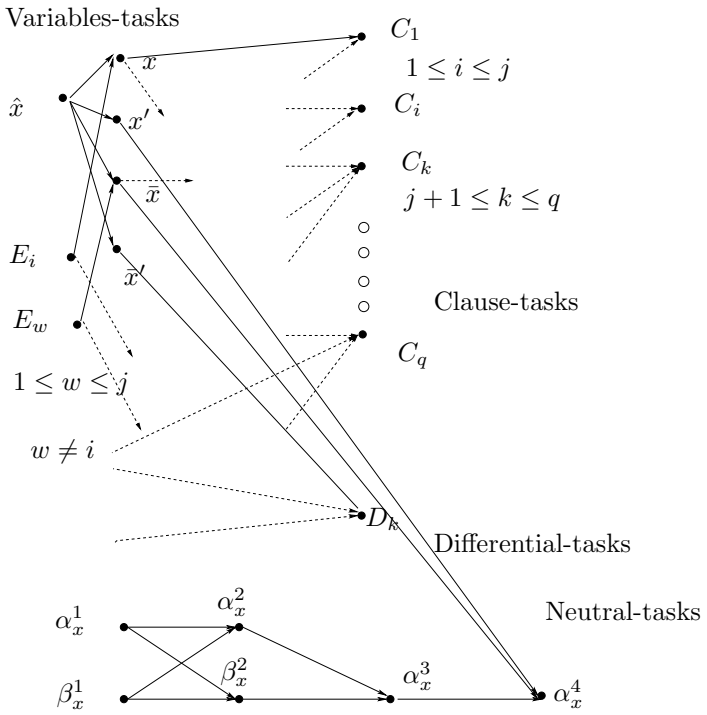
- For each variable $x \in \mathcal{V}$, we introduce six neutral-tasks $\alpha_x^1, \beta_x^1, \alpha_x^2, \beta_x^2, \alpha_x^3, \alpha_x^4$. The precedence constraints between these tasks are:
  $\alpha_x^1 \rightarrow \alpha_x^2, \beta_x^1 \rightarrow \beta_x^2, \beta_x^1 \rightarrow \alpha_x^2, \alpha_x^1 \rightarrow \beta_x^2, \alpha_x^2 \rightarrow \alpha_x^3, \beta_x^2 \rightarrow \alpha_x^3, \alpha_x^3 \rightarrow \alpha_x^4$.
  We also add the constraints:
  $\bar{x} \rightarrow \alpha_x^4$, $x' \rightarrow \alpha_x^4$.
- For every clause C_i, we introduce one clause-task C_i such that for every literal x occurring in C_i, we add the precedence constraint $x \rightarrow C_i$.
- For every clause of length three $(C_k, j+1 \leq k \leq q)$, we introduce one differential-task D_k such that for each literal x occurring in such a clause, we add the precedence constraints $\bar{x}' \rightarrow D_k$.
- For every clause of length two $(C_k, 1 \leq k \leq j)$, we introduce one differential-task E_k. We add the following precedence constraints: if x (resp. $\bar{x}$) occurs in C_k, then we add the constraint $E_k \rightarrow x$ (resp. $E_k \rightarrow \bar{x}$).

The above construction is illustrated in Figure 1. This transformation can be clearly computed in polynomial time.

Notation. In what follows, whenever we write: "$[y, z]$ is executed at time t on the cluster M", where y and z are tasks, then w.l.o.g. we will consider that these two tasks are executed simultaneously on M, the task y on processor P_1 and the task z on processor P_2 of M.

- Let us first assume that there is a truth assignment $I : \mathcal{V} \rightarrow \{0, 1\}$ such that each clause in $\mathcal{C}$ has exactly one true literal for the problem Π_1. Then we will prove that there is a schedule of length at most four.
  Let us construct this schedule:
  The variable-tasks $\hat{x}$, for all $x \in \mathcal{V}$, are executed at $t = 0$ on different clusters.
  - If the variable x is false, then $[x, x']$ is scheduled at $t = 1$ on the same cluster as $\hat{x}$ and the corresponding differential-task $E_k, 1 \leq k \leq j$. Otherwise, x and x' are scheduled at $t = 2$ on different clusters.
    The clause-tasks are executed at $t = 3$ on the same cluster as the only true literal.
  - If the variable $\bar{x}$ is false, then $[\bar{x}, \bar{x}']$ is scheduled at $t = 1$ on the same cluster as $\hat{x}$ and the corresponding differential-task $E_k, 1 \leq k \leq j$. Otherwise, $\bar{x}$ and $\bar{x}'$ are scheduled at $t = 2$ on two different clusters in the following way:
    - $\bar{x}$ is executed on the same cluster as its corresponding neutral-task α_x^3 and the associated clause-task C_k which is scheduled at $t = 3$;
    - $\bar{x}'$ is executed on a different cluster on which we also schedule at $t = 3$ the differential-task D_k. We can notice that at $t = 2$, two of the three predecessors of D_k finish their execution. This is always possible since we know that each clause contains only one true literal.

    The chain of neutral-tasks $\alpha_x^1, \beta_x^1, \alpha_x^2, \beta_x^2, \alpha_x^3, \alpha_x^4$ are scheduled on the same cluster as the variable-task x'.



	0	1	2	3	4
π_0^0	$\hat{x}$	x			
π_1^0	E_i	x'			
π_0^1	α_x^1	α_x^2	α_x^3	α_x^4	
π_1^1	β_x^1	β_x^2	$\bar{x}$	C_i	
π_0^2					
π_1^2			$\bar{x}'$	D_k	
π_0^3				C_1	
π_1^3				C_k	
π_0^4					
π_1^4	E_w				

The variable-tasks x is valued to "false"

The variable $\bar{x}$ is valued to "true"

FIGURE 1. The precedence graph corresponding to polynomial transformation $\Pi_1 \propto \bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{\max} = 4$.

For a clause of length three $(x \vee y \vee z)$, since this clause gets a truth assignment, we execute at $t = 1$ (resp. at $t = 2$) the two variables (resp. an unique variable) which are valued as false (resp. true).
W.l.o.g., we suppose that the variable x is valued as true and the variables y and z are valued as false. We execute the tasks (resp. the task) $\bar{y}$ and $\bar{z}$ (resp. $\bar{x}$) at $t = 2$ (resp. at $t = 1$) on different clusters as $\hat{y}$ and $\hat{z}$ (resp. on the same cluster as $\hat{x}$). Moreover, the differential-task E_k, where the variable $\bar{x}$ occurs, is executed at $t = 0$ on the same cluster as $\hat{x}$.
In the same way, the two differential-tasks $E_{k'}$ and $E_{k''}$, where the variable y and z occur, are executed on the same cluster as $\hat{y}$ and $\hat{z}$.
The tasks y' and z' are scheduled at $t = 1$ (resp. x' at $t = 2$) on the same cluster as $\hat{y}$ and $\hat{z}$ (resp. on a different cluster as $\hat{x}$). The tasks $\bar{y}'$ and $\bar{z}'$ are executed at $t = 2$ on the same cluster as the differential-task D_k associated to the clause $(x \vee y \vee z)$ and the task $\bar{x}'$ is scheduled at $t = 1$ on the same cluster as x.
For the clause of length two, the schedule is constructed in the similar way.
The above way of scheduling the tasks preserves the precedence constraints and the communication delays and gives a schedule of length four, whenever there is a truth assignment with exactly one true literal per clause.

- Conversely, suppose now that there is a schedule of length at most four. We will prove that there is an assignment $I : \mathcal{V} \to \{0, 1\}$ such that every clause has exactly one true literal.
  We start by making four essential observations.
  We notice that in every feasible schedule of length at most four:

  1. The neutral-tasks for all $x \in \mathcal{V}$ start their execution the earliest possible and have to be executed on the same cluster.
  2. In order to get a feasible schedule of length four, the variable-task $\hat{x}$ associated to the variable x (it is true for all variables) are scheduled at $t = 0$. Indeed, we suppose that $\hat{x}$ is executed at $t = 1$. The successors of the variable-task $\hat{x}$, $x, x', \bar{x}$ and $\bar{x}'$ must be executed on the same cluster between $[2, 4]$, and so the successors of these tasks have a starting time at $t = 4$, impossible. Moreover, the clause-tasks $C_i, 1 \leq i \leq q$ are executed at $t = 3$. Indeed, the clause-tasks $C_i, j + 1 \leq i \leq q$, associated to the clauses of length three, admit three predecessors whose cannot have starting time before $t = 1$. Therefore, theses clause-tasks must be executed at $t = 3$.
     The two predecessors of the clause-tasks $C_i, 1 \leq i \leq j$, associated to the clauses of length two, admit three predecessors, and one of these two predecessors of the clause-tasks C_i must be executed at least $t = 2$. Thus, the clause-tasks $C_i, 1 \leq i \leq j$ must be executed at $t = 3$.

3. The differential-task D_k corresponding to the clause-task C_k, $j+1 \leq k \leq q$, must be executed at $t = 3$. Indeed, each differential-task D_k admits three predecessors whose starting time is at least $t = 1$.
4. If the differential-task E_k, associated to the clause $C_k, 1 \leq k \leq j$ is executed at $t = 1$, the starting time of a clause-task or a differential-task $D_{k'}$ will be at $t = 4$.

Lemma 3.2. *In any valid schedule of length at most four the tasks $\bar{x}$ and x' (resp. $\bar{x}'$ and x) cannot be scheduled at the same time.*

Proof.

1. Let us assume that the tasks $\bar{x}$ and x' are scheduled at the same time:
   (a) The tasks $\bar{x}$ and x' are both predecessors of α_x^4 which must be scheduled at $t = 3$. But, this is not possible since α_x^4 is also preceded by α_x^3 which must also start at $t = 2$ and we have only two processors per cluster. Thus, $\bar{x}$ and x' cannot start simultaneously at $t = 2$.
   (b) The tasks $\bar{x}$ and x' cannot start at $t = 1$ (see 2(b)i).
2. Let us now assume that x and $\bar{x}'$ are scheduled at the same time. If they start at:
   (a) $t = 1$, then $\bar{x}$ and x' must be executed at $t = 2$. This is not possible, see 1a.
   (b) $t = 2$, then x and $\bar{x}'$ must be executed on different clusters. We have to examine the following two cases:
      (i) If x' is executed at $t = 1$. W.l.o.g. we may assume that x is an element of a clause of the form $(x \vee y \vee z)$. Then, the tasks y and z are executed at $t = 1$.
      Now, if the tasks $\bar{y}'$ and $\bar{x}'$ are executed simultaneously on the same cluster at $t = 2$, then the beginning of execution of $\bar{z}'$ must be at $t = 1$ on the same cluster as z. But in this case, $\bar{z}$ and z' are necessarily scheduled at $t = 2$, impossible from 1a. Using the same arguments $\bar{x}'$ and $\bar{z}'$ cannot be executed at $t = 2$ on the same cluster.
      It now remains to examine the case where the tasks $\bar{y}'$ and $\bar{z}'$ are executed at $t = 1$ on the same cluster as y and z. In this case, $\bar{y}$, y' and $\bar{z}$, z' are scheduled at $t = 2$, impossible from 1a.
      (ii) If $\bar{x}$ is executed at $t = 1$ and the tasks x, x' and $\bar{x}'$ at $t = 2$, then with the same type of arguments as in 2(b)i, it is easy to see that the schedule is not feasible.

In conclusion, the tasks $\bar{x}$ and x' (resp. $\bar{x}'$ and x) cannot be scheduled simultaneously. □

Lemma 3.3. *In any valid schedule of length at most four the tasks $\bar{x}$ and x cannot be executed simultaneously.*

Proof.

1. We suppose that the tasks x and $\bar{x}$ are executed at $t = 1$. We know that the clause $(x \vee \bar{x})$ cannot exist in any instance of Π_1 (see the definition

of the problem). Hence, $\bar{x}$ and x have always three predecessors: two differential-tasks E_i and E_j, with $i \neq j$, and the corresponding variable-task $\hat{x}$. Given that there is only two processors per cluster, the tasks x and $\bar{x}$ cannot be scheduled at $t = 1$.

2. We suppose that x and $\bar{x}$ are executed at $t = 2$. Two cases have to be examined:
   (a) Let us consider that x and $\bar{x}$ are executed at $t = 2$ on the same cluster.
   The chain of neutral-tasks associated to the variable x must be executed consecutively on the same cluster. Given that $\bar{x}$ is a predecessor of α_x^4 and $\bar{x}$ is processed at $t = 2$, we get that $\bar{x}$ must be executed on the same cluster as α_x^4. Thus, we have three tasks x, α_x^3 and $\bar{x}$ that should be processed at $t = 2$. Hence, the tasks x and $\bar{x}$ cannot be executed at $t = 2$ on the same cluster.
   (b) Let us now assume that the two tasks are executed at $t = 2$ on different clusters. In this case, it is clear that x' is necessarily executed at $t = 1$ (see Lem. 3.2):
   In order to clarify the presentation, we rename the variables of the logic formula in the following way: given that every variable occurs exactly once in a clause of length three, we rewrite the clauses of length three in the following form $(x_0 \vee x_1 \vee x_2) \wedge (x_3 \vee x_4 \vee x_5) \wedge \ldots \wedge (x_{n-3} \vee x_{n-2} \vee x_{n-1})$ with $x_i \neq x_j$ for all $i, j \in \{0, ..., n-1\}$. W.l.o.g. we assume in the following that $x = x_0$ and $\bar{x} = \bar{x}_0$. In order to prove that we cannot assign x_0 and $\bar{x}_0$ at $t = 2$, we will show first, that the assignment of x_0 and $\bar{x}_0$ at $t = 2$, implies a unique assignment for "some" literals, that we will call *critical literals*, (below we precise the way of identifying these literals) appearing to the set of clauses of length two and that this assignment implies a schedule of length greater than four, contradicting in this way the assumption. Before explaining the assignment of the literals, we introduce a method for the enumeration of the literals appearing in the set of clauses of length two.
   For every instance of Π_1, we consider the variable x_0. We know by the definition of Π_1 that the variable x_0 occurs two times in the set of clauses of length two: one time unnegated (in the form of literal x_0) and one time negated (in the form of literal $\bar{x}_0$). Now, we will explain a method to enumerate the *critical literals*:
   We consider the variable x_0. Let $(x_0 \vee \bar{x}_{k_1})$ be the clause of length two where the literal x_0 appears, and $(x_{k_2} \vee \bar{x}_0)$ the clause where the literal $\bar{x}_0$ appears.
   There are two cases to be considered:
      (i) if $k_1 = k_2$, then $x_0, \bar{x}_{k_1}, x_{k_1}, \bar{x}_0$ are the *critical literals*.
      (ii) if $k_1 \neq k_2, k_i \in \{1, n-1\}, i = \{1, 2\}$, then we continue. The literal $\bar{x}_{k_2}$ occurs also in a clause of length two denoted by $(x_{k_3} \vee \bar{x}_{k_2})$ such that $k_2 \neq k_3, k_i \in \{1, n-1\}, i = \{2, 3\}$.

In the same way, we know that the literal x_{k_1} occurs in a clause of length two denoted by $(x_{k_1} \vee \bar{x}_{k_4})$ such that $k_1 \neq k_4, k_i \in \{1, n-1\}, i = \{1, 3\}$. Up to now the *critical literals* are $x_0, \bar{x}_0, x_{k_1}, \bar{x}_{k_1}, x_{k_2}, \bar{x}_{k_2}, x_{k_3}$ and $\bar{x}_{k_4}$. We continue in this way until finding the first clause not containing a "new" variable. The following claim shows that there is always such a clause. The set of *critical literals* will contain all the literals encountered using the above described procedure.

Claim 3.4. During the procedure of enumeration of *critical literals* given above, we will always find a clause $(x_{k_j} \vee \bar{x}_{k_{j'}})$ with $k_j \neq k_{j'}$ such that the variables x_{k_j} and $x_{k_{j'}}$ have been already enumerated.

Proof. Let n be the number of variables in the logic formula. Thus, there are n clauses of length two and $2n$ literals.
Let us assume that the $(n-1)$ first clauses of length two that we examine by applying the procedure of enumeration do not verify the assumption of the claim. In this case, we have $(2n-2)$ *critical literals*. It remains only 2 literals to enumerate. We know that a clause of the form $(x_m \vee \bar{x}_m)$ cannot exist in any instance of Π_1 and consequently we can conclude that the remaining clause contains two literals that correspond to two variables already encountered. □

Now, we will prove that the assumption about the assignment of the tasks x_0 and $\bar{x}_0$ at $t = 2$ is false. For this, we show that the existence of a schedule of length at most four implies a unique way for the assignment of the tasks that correspond to the *critical literals*.
Two cases have to be taken into account:

(i) $\exists k \backslash (x_0 \vee \bar{x}_k) \wedge (x_k \vee \bar{x}_0)$. We suppose first that $\bar{x}_k$ is executed at $t = 2$. In the precedence graph for each variable x, there exists a chain of neutral-tasks of length four: $\alpha_x^1 \rightarrow \alpha_x^2 \rightarrow \alpha_x^3 \rightarrow \alpha_x^4$. In any schedule of length four, these tasks must be processed consecutively on the same cluster.
Given that $\bar{x}_k$ is a predecessor of $\alpha_{x_k}^4$ and $\bar{x}_k$ is processed at $t = 2$, then $\bar{x}_k$ must be executed on the same cluster as $\alpha_{x_k}^4$. moreover, by construction, we know that the tasks x_0 and $\bar{x}_k$ are both predecessors of the clause-task where the variables associated to the variable-tasks occurred, and the clause-task must be executed at $t = 3$ on the same cluster as the variable-task $\alpha_{x_k}^4$. So, three tasks, the tasks $x_0, \alpha_{x_k}^3$ and $\bar{x}_k$ must be processed on the same cluster at $t = 2$ in order to respect the schedule of length four. But, our model considers two processors per clause and so the schedule is not feasible.
Let us now show that x_k cannot be executed at $t = 2$. For the same reasons as before $\bar{x}_0$ must be executed (by the assumption at $t = 2$) on the same cluster as the neutral-tasks $\alpha_{x_0}^1 \rightarrow \alpha_{x_0}^2$

$\rightarrow \alpha^3_{x_0} \rightarrow \alpha^4_{x_0}$. But x_k and $\bar{x}_0$ are both predecessors of the clause-task that corresponds to the clause $(x_k \vee \bar{x}_0)$ and so, they must be executed on the same cluster. Again three tasks, the tasks $x_k, \bar{x}_0$ and $\alpha^3_{x_0}$ must be executed at $t = 2$ on the same cluster, a contradiction to the feasibility of schedule.

(ii) We consider the partial logic formula composed only by the clauses of length two containing the *critical literals*: $(x_0 \vee \bar{x}_{k_1}) \wedge (x_{k_2} \vee \bar{x}_0) \wedge (x_{k_1} \vee \bar{x}_{k_3}) \wedge (\bar{x}_{k_2} \vee x_{k_4}) \wedge \ldots \wedge (x_{k_i} \vee \bar{x}_{k_j}) \wedge \ldots \wedge (x_{k_{j'}} \vee \bar{x}_{k_{i'}}) \vee \ldots \vee (x_{k_j} \vee \bar{x}_{k_{j'}})$, with $k_i \neq k_{i'}$ and $k_i \neq k_j, k_{i'} \neq k_{j'}$.
For the same reason as previously, *i.e.* because of the existence of the neutral-tasks, in any feasible schedule of length at most four, the tasks x_0 and $\bar{x}_0$ cannot be executed on the same cluster at $t = 2$.
It remains now to prove that x_0 and $\bar{x}_0$ cannot be executed at $t = 2$ on different clusters. Consider the literal x_0. We know by the definition of the problem Π_1 that x_0 occurs one time in the clause $(x_0 \vee \bar{x}_{k_1})$, with $k_1 \in \{1, 2, \ldots, n-1\}$.
Then, we must executed at $t = 3$ on the same cluster as the two clause-tasks where the literal x_0 occurs. Thus, the variable-task $\bar{x}_{k_1}$ cannot be scheduled at $t = 2$, because $\bar{x}_{k_1}$ have two successors, the variable-task $\alpha^1_{k_1}$ and the clause-tasks where the literal associated to the variable-task x_0 occurs. Therefore, the variable-task $\bar{x}_{k_1}$ must executed at $t = 1$.
The variable-task x_{k_1} cannot be executed at $t = 1$, since in the one hand the clause $(x_{k_1} \vee \bar{x}_{k_1})$ cannot be existed and the other hand the variable-task x_{k_1} admits three predecessors (the variable-task $\hat{x}_{k_1}$ and the two differential-tasks E where the literal x_{k_1} and $\bar{x}_{k_1}$ occur).
Moreover, the task $\bar{x}_0$ occurs one time in the clause $(\bar{x}_0 \vee x_{k_2})$ with $k_2 \in \{1, 2, \ldots, n-1\}$ and $k_1 \neq k_2$. The literal x_{k_2} (resp. $\bar{x}_{k_2}$) must be executed at $t = 1$ (resp. at $t = 2$).
We repeat the assignment for every *critical literal* x_{k_i}, $k_i \in \{1, 2, \ldots, n-1\}$.
By the claim, we know that the clause $(x_{k_j} \vee \bar{x}_{k_{j'}})$, with $k_j \neq k_{j'}$ there exists always. In any feasible schedule of length at most four, the variable-tasks $x_{k_j}, \bar{x}_{k_j}, x_{k_{j'}}$ and $\bar{x}_{k_{j'}}$ are necessarily allocated using the above described procedure. It is easy to see that the variable-tasks $\bar{x}_{k_{j'}}$ and x_{k_j} are executed at $t = 2$. Given that $\bar{x}_{k_{j'}}$, must be executed on the same cluster as its corresponding neutral-tasks, we obtain that the corresponding clause-task cannot start before $t = 4$, a contradiction.

Consequently, the tasks $\bar{x}$ and x cannot be scheduled at time $t = 2$.

In conclusion, tasks $\bar{x}$ and x cannot be scheduled at the same time. □

From these two lemmas, we obtain the following corollary:

Corollary 3.5. *In any feasible schedule of length at most four, the tasks $\bar{x}$ and $\bar{x}'$ (resp. x and x') must be executed simultaneously.*

Proof. From the precedence constraints, we have that the tasks $x,\ x',\ \bar{x},\ \bar{x}'$ must be scheduled at $t = 1$ or $t = 2$. From Lemmata 3.2 and 3.3 $t_x \neq t_{x'}$ and $t_{\bar{x}} \neq t_{x'}$, so $t_x = t_{\bar{x}}$ (t_x is the starting time of x). With the same argument we have $t_{x'} = t_{\bar{x}'}$. □

We can give now, the Lemma 3.6 which states that there are two possibilities for the execution of $\bar{x}$ and $\bar{x}'$ (resp. x and x'): either they are executed on the same cluster at $t = 1$ or on different clusters at $t = 2$.

Lemma 3.6. *In any feasible schedule of length at most four, the tasks $\bar{x}$ and $\bar{x}'$ (resp. x and x') are executed on the same cluster at $t = 1$, otherwise on different clusters at $t = 2$.*

Proof. If $t_x = t_{x'} = 1$, from the precedence constraints $\hat{x} \to x'$ and $\hat{x} \to x$ we deduce that x and x' are scheduled by the same cluster as $\hat{x}$. If $t_x = t_{x'} = 2$, from the precedence constraint $x' \to \alpha_x^4$, x' is scheduled by the same cluster as α_x^4, and thus x is scheduled by some other cluster. The same arguments hold for $\bar{x}$ and $\bar{x}'$. □

There exist two possibilities for scheduling $[\bar{x}, \bar{x}']$ and $[x, x']$: at $t = 1$ on the same cluster as the variable-task $\hat{x}$, or at $t = 2$ on different clusters (see Lem. 3.6).

- Define a literal x as "true" (resp. "false") if the corresponding variable-task x is processed at time $t = 2$ (resp. at $t = 1$).
- Define a literal $\bar{x}$ as "true" (resp. "false") if the corresponding variable-task $\bar{x}$ is processed at time $t = 2$ (resp. at $t = 1$).

Lemma 3.7. *If an instance π of the problem $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\max}$ has a schedule of length at most four, then the corresponding instance π^* of Π_1 has a truth assignment satisfying the logic formula and such that at each clause there is exactly one true literal.*

Proof. We suppose that there exists a schedule of length four. We will show that for each clause there exists exactly one literal which is processed at $t = 2$ and the other literal(s) at $t = 1$.

1. We consider a clause C of length two denoted w.l.o.g. by $(\bar{x} \vee y)$.
   (a) Assume that the tasks $\bar{x}$ and y are executed at $t = 2$ on the same cluster. Task $\bar{x}$ has α_x^4 for successor and thus it is processed by the same cluster as α_x^4. Hence, $\alpha_x^3,\ \bar{x},\ y$ are scheduled at $t = 2$ by the same cluster, a contradiction.
   (b) We suppose now that $\bar{x}$ and y are executed at $t = 1$ on the same cluster. For every clause $(\bar{x} \vee y)$ of length two, there is a differential-task E and the following precedence constraints: $E \to \bar{x}$ and $E \to y$. The task $\bar{x}$ (resp. y) has also a predecessor $\hat{x}$ (resp. $\hat{y}$). Given the

precedence constraints, the task $\hat{x}$ (resp. $\hat{y}$) must be processed at $t = 0$ on the same cluster as $\bar{x}$ (resp. y). So, at $t = 0$, we have three tasks, $\hat{x}, \hat{y}$ and a differential-task E, and only two processors in the same cluster. Consequently, the tasks $\bar{x}$ and y cannot be executed on the same cluster at $t = 1$. Notice also that $\bar{x}$ and y cannot be executed at $t = 1$ on different clusters, because they have a common predecessor, the differential-task E. So, only one of the tasks $\bar{x}$ and y can be executed at $t = 1$.

In conclusion, if we execute $\bar{x}$ (resp. y) at $t = 1$, then according to Lemma 3.6, $\bar{x}'$ (resp. y') must be executed on the same cluster as $\bar{x}$ (resp. as y).

From the definitions given above, $\bar{x}$ (resp. y) is defined as "false" (resp. as "true") because the corresponding variable-task $\bar{x}$ (resp. y) is processed at $t = 1$ (resp. at $t = 2$). So we have exactly one true literal.

2. We consider now, a clause C_k of length three, with $j+1 \leq k \leq q$ denoted by $(x \vee y \vee z)$. Clearly, the variable-tasks x, y, z cannot be scheduled at $t = 0$ (resp. $t = 2$) because of the precedence constraints. We suppose that the three variable-tasks x, y and z are scheduled at $t = 1$. They can be executed only on different clusters. From Lemma 3.2, the variable-tasks x and $\bar{x}$ cannot be executed at the same time. Therefore, the variable-task $\bar{x}', \bar{y}'$ and $\bar{z}'$ must be executed at least at $t = 2$. But, these variable-tasks have a common successor, the differential-task D_k, associated to the clause C_k, whose starting time is $t = 4$, impossible.

For all the clauses in the logic formula, there is exactly one true literal. □

This concludes the proof of Theorem 3.1 □

Corollary 3.8. *There is no polynomial-time algorithm for the problem* $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\max}$ *with performance bound smaller than* $\frac{5}{4}$ *unless* $\mathcal{P} = \mathcal{NP}$.

Proof. The proof of Corollary 3.8 is an immediate consequence of the Impossibility Theorem (see [3, 4]). □

4. A polynomial time algorithm for $C_{\max} = 3$

Theorem 4.1. *The problem of deciding whether an instance of* $\bar{P}(P2)|\text{prec}; p_i = 1; (c_{ij}, \epsilon_{ij}) = (1, 0)|C_{\max}$ *has a schedule of length at most three is solvable in polynomial time.*

Proof. Given an arbitrary instance of the problem $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\max} = 3$, we first check whether some obviously necessary conditions hold. First, we check that the graph does not contain any path of length more than four.

We denote by $\delta^-(x)$ (resp. $\delta^+(x)$) the number of predecessors (resp. of successors) of x in the graph G.

Let $G = (V, E)$ be the initial graph.

We divide the set of vertices V into four classes:

1. $\mathcal{S} = \{v \in V \backslash \delta^-(v) = 0 \text{ and } \delta^+(v) \neq 0\}$;
2. $\mathcal{Z} = \{v \in V \backslash \delta^+(v) = 0 \text{ and } \delta^-(v) \neq 0\}$;
3. $\mathcal{I} = \{v \in V \backslash \delta^-(v) = \delta^+(v) = 0\}$;
4. $\mathcal{X} = V - (\mathcal{S} \bigcup \mathcal{Z} \bigcup \mathcal{I})$.

For every task $x_i \in \mathcal{X}$, we define a set $K_i = \Gamma^-(x_i) \bigcup \{x_i\} \bigcup \Gamma^+(x_i)$ where $\Gamma^-(x_i)$ (resp. $\Gamma^+(x_i)$) denotes the set of predecessors (resp. successors) of the task x_i.

Now we consider $R \subseteq V \times V$ such that $(x_i, x_j) \in R$ if and only if $x_i \in K_i, x_j \in K_j$ and $K_i \bigcap K_j \neq \varnothing$, and let R^* be its transitive closure. R^* is an equivalence relation on $\bigcup K_j = W$. An equivalence class of R^* will be called *group of tasks* in the sequel. The tasks which are not elements of one of the equivalence classes W_i, are elements of one the sets $\mathcal{S}$, $\mathcal{Z}$ or $\mathcal{I}$. Given that the tasks of $\mathcal{S}$ and $\mathcal{I}$ (resp. $\mathcal{Z}$) do not belong to a path of length 3, we can execute them at time 0 (resp. at time 2).

Each *group of tasks* constitutes a set of tasks that have to be executed by the same cluster in order to yield a schedule within three time units. This is a direct implication of the fact that each task in $\mathcal{X}$ has to be executed at $t = 1$ and on the same cluster with its predecessors and its successors. Consequently the following condition hold: there is no feasible schedule within three time units, if there is a *group of tasks* W_i such that $|W_i \bigcap \mathcal{S}| > 2$, or $|W_i \bigcap \mathcal{Z}| > 2$, or $|W_i \bigcap \mathcal{X}| > 2$. □

5. Conclusions and future work

In this paper, we first proved that the problem of deciding whether an instance of $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\max}$ has a schedule of length at most four is $\mathcal{NP}$-complete. This result is to be compared with the result of [6], which states that $\bar{P}|\text{prec}; c_{ij} = 1; p_i = 1|C_{\max} = 6$ is $\mathcal{NP}$-complete. Our result implies that there is no $\rho-$approximation algorithm with $\rho < \frac{5}{4}$, unless $\mathcal{P} = \mathcal{NP}$. We also established that the problem of deciding whether an instance of $\bar{P}(P2)|\text{prec}; (c_{ij}, \epsilon_{ij}) = (1, 0); p_i = 1|C_{\max}$ has a schedule of length at most three is solvable in polynomial time. An interesting question for further research is to find an approximation algorithm with performance guarantee better than the trivial bound of two.

References

[1] E. Bampis, R. Giroudeau and J.C. König, Using duplication for multiprocessor scheduling problem with hierarchical communications. *Parallel Process. Lett.* **10** (2000) 133-140.

[2] B. Chen, C.N. Potts and G.J. Woeginger, *A review of machine scheduling: Complexity, algorithms and approximability*, Technical Report Woe-29. TU Graz (1998).
[3] Ph. Chrétienne, E.J. Coffman Jr., J.K. Lenstra and Z. Liu, *Scheduling Theory and its Applications*. Wiley (1995).
[4] M.R. Garey and D.S. Johnson, *Computers and Intractability, a Guide to the Theory of $\mathcal{NP}$-Completeness*. Freeman (1979).
[5] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Optimization and approximation in deterministics sequencing and scheduling theory: A survey. *Ann. Discrete Math.* **5** (1979) 287-326.
[6] J.A. Hoogeveen, J.K. Lenstra and B. Veltman. Three, four, Five, six, or the complexity of scheduling with communication delays. *Oper. Res. Lett.* **16** (1994) 129-137.

to access this journal online:
www.edpsciences.org